

Continuous-Time Visual-Inertial Trajectory Estimation with Event Cameras

Elias Mueggler, Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza

Abstract—In contrast to traditional cameras, which output images at a fixed rate, event cameras have independent pixels that output asynchronous pixel-level brightness changes with microsecond resolution. In this paper, we leverage a continuous-time framework to perform trajectory estimation by fusing visual data from a moving event camera with inertial data from an IMU. This framework allows direct integration of the asynchronous events with micro-second accuracy and the inertial measurements at high frequency. The pose trajectory is approximated by a smooth curve in the space of rigid-body motions using cubic splines. This formulation significantly reduces the number of variables in trajectory estimation problems. We evaluate our method on real data from several scenes and compare the results against ground truth from a motion-capture system. We show superior performance of the proposed technique compared to non-batch event-based algorithms. We also show that both the map orientation and scale can be recovered accurately by fusing events and inertial data. To the best of our knowledge, this is the first work on visual-inertial fusion with event cameras using a continuous-time framework.

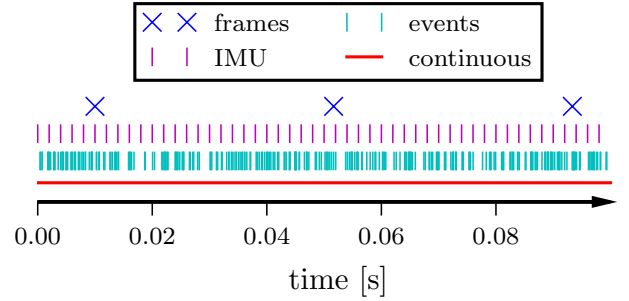


Fig. 1: While the frames and inertial measurements arrive at a constant frequency, events are transmitted asynchronously and at much higher frequency. We model the camera trajectory as continuous in time, which allows direct integration of all measurements using their precise timestamps.

I. INTRODUCTION

STANDARD frame-based CMOS cameras operate at fixed frame rates, sending entire images at constant time intervals that are selected based on the considered application [1]. Contrary to standard cameras, where pixels are acquired at regular time intervals (e.g., global shutter or rolling shutter), event cameras, such as Dynamic Vision Sensors (DVS [2] or DAVIS [3]), have asynchronous, independent pixels: each pixel of an event camera immediately triggers an *event* whenever it detects a brightness change in the scene (Fig. 1). The temporal resolution of such events is in the order of *micro*-seconds. It is only the sign of these changes that is transmitted, which is binary (increase or decrease of local brightness). Because the output it produces—an event stream—is fundamentally different from video streams of standard cameras, new algorithms are required to deal with these data.

We aim to use event cameras for ego-motion estimation. The approach provided by traditional visual-odometry frameworks, which estimate the camera pose at discrete times (naturally, the times the images are acquired), is no longer appropriate for event cameras, mainly due to two issues. First, a single event does not contain enough information to estimate the sensor pose given by the six degrees of freedom (DOF) of a calibrated camera. Additionally, it is not appropriate to simply consider several events for determining the pose

using standard computer-vision techniques, such as PnP [4], because the events typically all have different timestamps, and so the resulting pose will not correspond to any particular time. Second, an event camera can easily transmit up to several million events per second, and, therefore, it can become intractable to estimate the pose of the event camera at the discrete times of all events due to the rapidly growing size of the state vector needed to represent all such poses.

To tackle the above-mentioned issues, we adopt a continuous-time framework [5]. Regarding the first issue, an explicit continuous temporal model is a natural representation of the pose trajectory $T(t)$ of the event camera since it unambiguously relates each event, occurring at time t_k , with its corresponding pose, $T(t_k)$. To solve the second issue, the trajectory is described by a smooth parametric model, with significantly fewer parameters than events, hence achieving state space size reduction and computational efficiency. For example, to remove unnecessary states for the estimation of the trajectory of dynamic objects, [6] proposed to use cubic splines, reporting state-space size compression of 70–90%. Cubic splines [7] or, in more general, Wavelets [8] are common basis functions for continuous-time trajectories. The continuous-time framework was also motivated to allow data fusion of multiple sensors working at different rates and to enable increased temporal resolution [6]. The framework has been applied to camera-IMU fusion [7], [5], rolling-shutter cameras [5], actuated lidar [9], and RGB-D rolling shutter cameras [10].

The authors are with the Robotics and Perception Group, University of Zurich, Switzerland—<http://rpg.ifi.uzh.ch>. This research was supported by the National Centre of Competence in Research (NCCR) Robotics, the Qualcomm Innovation Fellowship, and the UZH Forschungskredit

Contribution

The use of a continuous time framework for ego motion estimation with event cameras was first introduced in our previous work [11]. In the present paper, we show instead how the same framework can be extended to incorporate inertial data, yielding a principled optimization that fuses event data with inertial observations. We show that this framework allows us to produce more accurate trajectories than with visual data alone and to estimate the absolute scale and orientation (alignment with respect to gravity). Additionally, while [11] was limited to highly-textured polygonal maps, we extend the approach to work on natural scenes using point-cloud maps. Finally, we also show that our approach can be used to refine the poses estimated by an event-based visual odometry method [12].

The paper is organized as follows: Section II briefly introduces the principle of operation of event cameras, Section III reviews previous work on ego-motion estimation with event cameras, Sections IV to VI present our method for continuous time trajectory optimization using visual-inertial event data fusion, Section VII presents the experiments carried out using the event camera to track two types of maps (point-based and line-based), Section VIII discusses the results, and Section IX draws final conclusions.

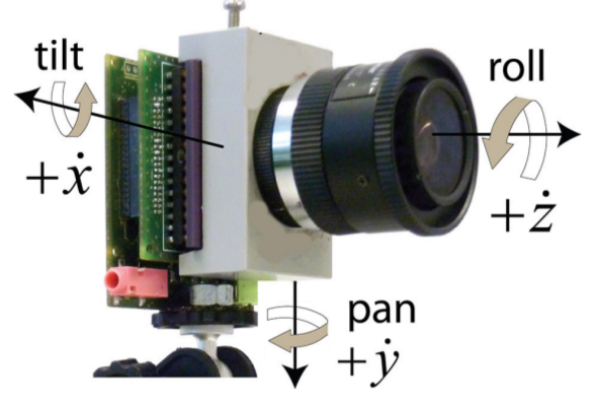
II. EVENT CAMERAS

Standard cameras acquire frames (i.e., images) at fixed rates. On the other hand, event cameras such as the DAVIS [3] (Fig. 2) have independent pixels that output brightness changes (called “events”) asynchronously, at the time they occur. Specifically, if $L(\mathbf{u}, t) \doteq \log I(\mathbf{u}, t)$ is the logarithmic brightness or intensity at pixel $\mathbf{u} = (x, y)^\top$ in the image plane, the DAVIS generates an event $e_k \doteq \langle x_k, y_k, t_k, p_k \rangle$ if the change in logarithmic brightness at pixel $\mathbf{u}_k = (x_k, y_k)^\top$ reaches a threshold C (typically 10-15% relative brightness change):

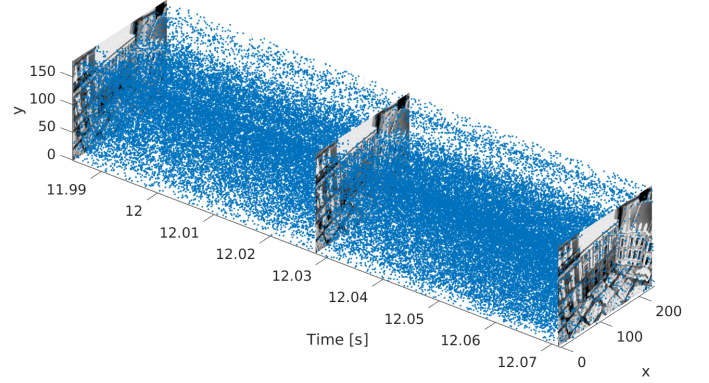
$$\Delta L(\mathbf{u}_k, t_k) \doteq L(\mathbf{u}_k, t_k) - L(\mathbf{u}_k, t_k - \Delta t) = p_k C, \quad (1)$$

where t_k is the timestamp of the event, Δt is the time since the previous event at the same pixel \mathbf{u}_k and $p_k = \pm 1$ is the polarity of the event (the sign of the brightness change). Events are timestamped and transmitted asynchronously at the time they occur using a sophisticated digital circuitry.

Event cameras have the same optics as traditional perspective cameras, therefore, standard camera models (e.g., pinhole) still apply. In this work, we use the DAVIS 240C [3] that provides events and global-shutter images from the same physical pixels. In addition to the events and images, it contains a synchronized IMU. This work solely uses the images for camera calibration, initialization and visualization purposes. The sensor’s spatial resolution is 240×180 pixels and it is connected via USB. A visualization of the output of the DAVIS is shown in Fig. 2b. An additional advantage of the DAVIS is its very high dynamic range of 130 dB (compared to 60 dB of high quality traditional image sensors).



(a) The DAVIS sensor (Figure adapted from [13]).



(b) Visualization of the event output of a DAVIS in space-time. Blue dots mark individual asynchronous events. The polarity of the events is not shown.

Fig. 2: The DAVIS camera and visualization of its output.

III. RELATED WORK: EGO-MOTION ESTIMATION WITH EVENT CAMERAS

A particle-filter approach for robot self-localization using the DVS was introduced in [14] and later extended to SLAM in [15]. However, the system was limited to planar motions and planar scenes parallel to the plane of motion, and scenes consisted of B&W line patterns.

In several works, conventional vision sensors have been attached to the event camera to simplify the ego-motion estimation problem. For example, [16] proposed an event-based probabilistic framework to update the relative pose of a DVS with respect to the last frame of an attached standard camera. The 3-D SLAM system in [17] relied on a frame-based RGB-D camera attached to the DVS to provide depth estimation, and thus build a voxel grid map that was used for pose tracking. The system in [18] used the intensity images from the DAVIS camera to detect features that were tracked using the events and were then fed into a 3-D visual odometry pipeline.

Robot localization in 6-DOF with respect to a map of B&W lines was demonstrated using a DVS, without additional sensing, during high-speed maneuvers of a quadrotor [19], where rotational speeds of up to $1,200^\circ/\text{s}$ were measured. In natural scenes, [20] presented a probabilistic filter to track

high-speed 6-DOF motions with respect to a map containing both depth and brightness information.

A system with two probabilistic filters operating in parallel was presented in [21] to estimate the rotational motion of a DVS and reconstruct the scene brightness in a high-resolution panorama. The system was extended in [22] using three filters that operated in parallel to estimate the 6-DOF pose of the event camera, and the depth and brightness of the scene.

More recently, [12] presented a geometric parallel-tracking-and-mapping approach for 6-DOF pose estimation and 3-D reconstruction with an event camera in natural scenes.

All previous methods operate in an event-by-event basis producing estimates of the event camera pose in a discrete, filter-like manner. This paper leverages a continuous-time representation of the trajectory of the event camera to couple the estimated poses in a tractable batch optimization that also allows to fuse event and inertial data.

IV. CONTINUOUS-TIME TRAJECTORIES

Traditional visual odometry and SLAM formulations use a discrete-time approach, i.e., the camera pose is calculated at the time the image was acquired. Recent works have shown that, for high-frequency data, a continuous-time formulation is preferable to keep the size of the optimization problem bounded [7], [5]. Temporal basis functions, such as B-splines, were proposed for camera-IMU calibration, where the frequencies of the two sensor modalities differ by an order of magnitude. While previous approaches use continuous-time representations mainly to reduce the computational complexity, in the case of an event-based sensor this representation is required to cope with the asynchronous nature of the events. Unlike a standard camera image, an event does not carry enough information to estimate the sensor pose by itself. A continuous-time trajectory can be evaluated at any time, in particular at each event's and inertial measurement's timestamp, yielding a well-defined pose for every event and well-defined derivatives. Thus, our method is not only computationally effective, but it is also necessary for a proper formulation.

A. Camera Pose Transformations

Following [5], we represent Euclidean space transformations between finite cameras [23, p. 157] by means of 4×4 matrices of the form

$$\mathbf{T}_{b,a} = \begin{bmatrix} \mathbf{R}_{b,a} & \mathbf{t}_a \\ \mathbf{0}^\top & 1 \end{bmatrix}, \quad (2)$$

where $\mathbf{R} \in SO(3)$ (the rotation group) and $\mathbf{t} \in \mathbb{R}^3$ are the rotational and translational components of the rigid-body motion, respectively. In homogeneous coordinates, a 3-D point in frame a is mapped to a point in frame b by the change of coordinates $\mathbf{X}_b \sim \mathbf{T}_{b,a}\mathbf{X}_a$, where \sim means equality up to a non-zero scale factor. Transformations (2) form the special Euclidean group $SE(3)$ [24, p. 30], which has the structure of both a group and a differentiable manifold, i.e., a Lie group. A curve on $SE(3)$ physically represents the motion of a rigid body, e.g., the event camera. The tangent space of $SE(3)$ at the identity is $se(3)$, which has the structure of a Lie algebra.

It corresponds to the space of *twists*, represented by 4×4 matrices of the form

$$\hat{\xi} = \begin{bmatrix} \hat{\omega} & \mathbf{v} \\ \mathbf{0}^\top & 0 \end{bmatrix}, \quad (3)$$

where $\mathbf{v} \in \mathbb{R}^3$ and $\hat{\omega}$ is the 3×3 skew-symmetric matrix representing the cross product: $\hat{\omega}\mathbf{b} = \omega \times \mathbf{b}$, $\forall \omega, \mathbf{b} \in \mathbb{R}^3$. Variables ω and \mathbf{v} physically represent the angular and linear velocity vectors of the moving sensor.

Based on the theory of Lie groups, the exponential map from $se(3)$ to $SE(3)$ can be defined, which gives the Euclidean transformation associated to a twist, $\mathbf{T} = \exp(\hat{\xi})$. The inverse of the exponential map is the logarithmic map $\hat{\xi} = \log(\mathbf{T})$. Moreover, every rigid-body motion $\mathbf{T} \in SE(3)$ can be represented in such an exponential parametrization, but the resulting twist may not be unique [24, p. 33]. However, to avoid this ambiguity, we adopt a local-chart approach (on the manifold $SE(3)$) by means of incremental rigid-body motions ($\mathbf{T} = \exp(\hat{\xi})$ with small matrix norm $\|\hat{\xi}\|$) given by the relative transformation between two nearby poses along the trajectory of the event camera (see (8)). In addition, this parametrization is free from singularities. Closed-form formulas for the the exp and log maps are given in [24].

The vee operator $[\cdot]^\vee$, inverse of the lift operator $\hat{\cdot}$, maps a 3×3 skew-symmetric matrix to its corresponding vector,

$$\begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}^\vee = \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \quad (4)$$

We can “linearly” interpolate between two poses at the extremes of an interval $[0, \Delta t] \ni t$ using formula

$$\mathbf{T}_w(t) = \mathbf{T}_{w,a} \exp\left(t \frac{1}{\Delta t} \log(\mathbf{T}_{w,a}^{-1} \mathbf{T}_{w,b})\right). \quad (5)$$

B. Cubic Spline Camera Trajectories in $SE(3)$

We use B-splines to represent continuous-time trajectories in $SE(3)$ for several reasons: they (i) are smooth (C^2 continuity in case of cubic splines), (ii) have local support, (iii) have analytical derivatives and integrals, (iv) interpolate the pose at any point in time, thus enabling data fusion from both asynchronous and synchronous sensors with different rates.

The continuous trajectory of the event camera is parametrized by control camera poses $\mathbf{T}_{w,i}$ at times t_i , $i \in \{0, \dots, n\}$, where, according to the notation in (2), $\mathbf{T}_{w,i}$ is the transformation from the event camera frame at time t_i to a world frame (w). Due to the locality of the cubic B-spline basis, the value of the spline curve at any time t only depends on four control poses: for $t \in [t_i, t_{i+1}]$ such control poses occur at times $\{t_{i-1}, \dots, t_{i+2}\}$. Following the cumulative cubic B-splines formulation [5], we use one absolute pose, $\mathbf{T}_{w,i-1}$, and three incremental poses, parameterized by twists (3) $\hat{\xi}_q \equiv \Omega_q$ (local approach on $SE(3)$). More specifically, the spline trajectory is given by

$$\mathbf{T}_{w,s}(u(t)) \doteq \mathbf{T}_{w,i-1} \prod_{j=1}^3 \exp\left(\hat{\mathbf{B}}_j(u(t)) \Omega_{i+j-1}\right), \quad (6)$$

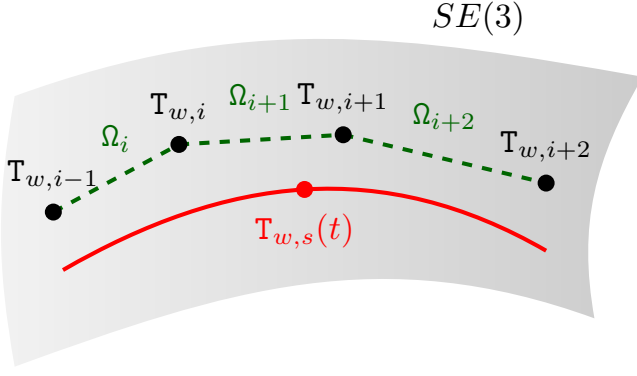


Fig. 3: Geometric interpretation of the cubic spline interpolation given by formula (6). The cumulative formulation uses one absolute control pose $T_{w,i-1}$ and three incremental control poses $\Omega_i, \Omega_{i+1}, \Omega_{i+2}$ to compute the interpolated pose $T_{w,s}$.

where, for simplicity, we assume that the control poses are uniformly spaced in time [5], at $t_i = i\Delta t$, thus $u(t) = (t - t_i)/\Delta t \in [0, 1)$ is used in the cumulative basis functions for the B-splines,

$$\tilde{\mathbf{B}}(u) = \mathbf{C} \begin{bmatrix} 1 \\ u \\ u^2 \\ u^3 \end{bmatrix}, \quad \mathbf{C} = \frac{1}{6} \begin{bmatrix} 6 & 0 & 0 & 0 \\ 5 & 3 & -3 & 1 \\ 1 & 3 & 3 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (7)$$

which are obtained from the matrix representation of the De Boor-Cox formula [25]. In (6), $\tilde{\mathbf{B}}_j$ is the j -th entry (0 based) of the cubic polynomial vector. The incremental pose from the frame at t_{q-1} to the frame at t_q in terms of world-referenced poses is encoded by the twist

$$\Omega_q = \log(T_{w,q-1}^{-1} T_{w,q}). \quad (8)$$

Figure 3 visualizes the evaluation of the pose $T_{w,s}(t) \equiv T_{w,s}(u(t))$ using one control pose $T_{w,i-1}$ and three incremental poses Ω .

The first and second temporal derivatives of the spline trajectory (6) are, using Newton's dot notation for differentiation,

$$\dot{T}_{w,s}(u) = T_{w,i-1} \begin{pmatrix} \dot{\mathbf{A}}_0 \mathbf{A}_1 \mathbf{A}_2 \\ + \mathbf{A}_0 \dot{\mathbf{A}}_1 \mathbf{A}_2 \\ + \mathbf{A}_0 \mathbf{A}_1 \dot{\mathbf{A}}_2 \end{pmatrix}, \quad (9)$$

$$\ddot{T}_{w,s}(u) = T_{w,i-1} \begin{pmatrix} \ddot{\mathbf{A}}_0 \mathbf{A}_1 \mathbf{A}_2 + \mathbf{A}_0 \ddot{\mathbf{A}}_1 \mathbf{A}_2 \\ + \mathbf{A}_0 \mathbf{A}_1 \ddot{\mathbf{A}}_2 + 2\dot{\mathbf{A}}_0 \dot{\mathbf{A}}_1 \mathbf{A}_2 \\ + 2\dot{\mathbf{A}}_0 \mathbf{A}_1 \dot{\mathbf{A}}_2 + 2\mathbf{A}_0 \dot{\mathbf{A}}_1 \dot{\mathbf{A}}_2 \end{pmatrix}, \quad (10)$$

respectively, where

$$\mathbf{A}_j \doteq \exp(\Omega_{i+j} \tilde{\mathbf{B}}(u)_{j+1}), \quad (11)$$

$$\dot{\mathbf{A}}_j = \mathbf{A}_j \Omega_{i+j} \dot{\tilde{\mathbf{B}}}(u)_{j+1}, \quad (12)$$

$$\ddot{\mathbf{A}}_j = \dot{\mathbf{A}}_j \Omega_{i+j} \dot{\tilde{\mathbf{B}}}(u)_{j+1} + \mathbf{A}_j \Omega_{i+j} \ddot{\tilde{\mathbf{B}}}(u)_{j+1}, \quad (13)$$

$$\dot{\tilde{\mathbf{B}}} = \frac{1}{\Delta t} \mathbf{C} \begin{bmatrix} 0 \\ 1 \\ 2u \\ 3u^2 \end{bmatrix}, \quad \ddot{\tilde{\mathbf{B}}} = \frac{1}{\Delta t^2} \mathbf{C} \begin{bmatrix} 0 \\ 0 \\ 2 \\ 6u \end{bmatrix}. \quad (14)$$

Analytical derivatives of $T_{w,s}(u)$ with respect to the control poses are provided in the supplementary material of [10].

C. Visual and Inertial Predictions

A continuous trajectory model allows us to compute the velocity and acceleration of the event camera at any time. These quantities can be compared against IMU measurements and the resulting mismatch can be used to refine the modeled trajectory. Similarly to [5], the predictions of the IMU measurements, in angular velocity ω and linear acceleration \mathbf{a} , are given by

$$\hat{\omega}(u) \doteq (\mathbf{R}_{w,s}^\top(u) \cdot \dot{\mathbf{R}}_{w,s}(u))^\vee + \mathbf{b}_\omega, \quad (15)$$

$$\hat{\mathbf{a}}(u) \doteq (\mathbf{R}_{w,s}^\top(u) \cdot (\ddot{\mathbf{s}}_w(u) + \mathbf{g}_w)) + \mathbf{b}_a, \quad (16)$$

where $\dot{\mathbf{R}}_{w,s}(u)$ and $\ddot{\mathbf{s}}_w(u)$ are the appropriate sub-matrices of $\dot{T}_{w,s}(u)$ and $\ddot{T}_{w,s}(u)$, respectively, \mathbf{b}_ω and \mathbf{b}_a are the gyroscope and accelerometer biases, and \mathbf{g}_w is the acceleration due to gravity in the world frame. The vee operator $^\vee$ is defined in (4).

V. MAP REPRESENTATION

To focus on the event-camera trajectory estimation problem, we assume that the map of the scene is given and is time invariant. Specifically, the map \mathcal{M} is either a set of 3-D points or 3-D line segments. We provide experiments using both geometric primitives.

In case of a map consisting of a set of points

$$\mathcal{M} = \{\mathbf{X}_i\}, \quad (17)$$

since events are caused by the apparent motion of edges, each 3-D point \mathbf{X}_i represents a scene edge. Given a 3×4 projection matrix \mathbf{P} modeling the perspective projection carried out by the event camera, the event coordinates are, in homogeneous coordinates, $\mathbf{u}_i \sim \mathbf{P}\mathbf{X}_i$.

In the case of lines, the map is

$$\mathcal{M} = \{\ell_j\}, \quad (18)$$

where each line segment ℓ_j is parametrized by its start and end points $\mathbf{X}_j^s, \mathbf{X}_j^e \in \mathbb{R}^3$. The lines of the map \mathcal{M} can be projected to the image plane by projecting the endpoints of the segments. The homogeneous coordinates of the projected line through the j -th segment are

$$l_j \sim (\mathbf{P}\mathbf{X}_j^s) \times (\mathbf{P}\mathbf{X}_j^e). \quad (19)$$

VI. TRAJECTORY OPTIMIZATION

In this section, we formulate the camera trajectory estimation problem from visual and initial data in a probabilistic framework and derive the maximum likelihood solution (Section VI-A). Then, to find a tractable solution, we reduce the dimensionality of the problem (Section VI-B) by using the parametrized cubic spline trajectory representation introduced in Section IV-B.

A. Probabilistic Approach

In general, the trajectory estimation problem over an interval $[0, T]$ can be cast in a probabilistic form [7], seeking an estimate of the joint posterior density $p(\mathbf{x}(t)|\mathcal{M}, \mathcal{Z})$ of the state $\mathbf{x}(t)$ (event camera trajectory) over the interval, given the map \mathcal{M} and the set of all visual-inertial measurements, $\mathcal{Z} = \mathcal{E} \cup \mathcal{W} \cup \mathcal{A}$, which consists of: events $\mathcal{E} \doteq \{\mathbf{e}_i\}_{i=1}^N$ (where $\mathbf{e}_k = (x_k, y_k)^\top$ is the event location at time t_k), angular velocities $\mathcal{W} \doteq \{\boldsymbol{\omega}_i\}_{i=1}^M$ and linear accelerations $\mathcal{A} \doteq \{\mathbf{a}_j\}_{j=1}^M$. Using Bayes' rule, and assuming that the map is independent of the event camera trajectory, we may rewrite the posterior as

$$p(\mathbf{x}(t) | \mathcal{M}, \mathcal{E}, \mathcal{W}, \mathcal{A}) \propto p(\mathbf{x}(t)) p(\mathcal{E}, \mathcal{W}, \mathcal{A} | \mathbf{x}(t), \mathcal{M}). \quad (20)$$

In the absence of prior belief for the state, $p(\mathbf{x}(t))$, the optimal trajectory is the one maximizing the likelihood $p(\mathcal{E}, \mathcal{W}, \mathcal{A} | \mathbf{x}(t), \mathcal{M})$. Assuming that the measurements $\mathcal{E}, \mathcal{W}, \mathcal{A}$ are independent of each other given the trajectory and the map, and using the fact that the inertial measurements do not depend on the map, the likelihood factorizes:

$$p(\mathcal{E}, \mathcal{W}, \mathcal{A} | \mathbf{x}(t), \mathcal{M}) = p(\mathcal{E} | \mathbf{x}(t), \mathcal{M}) p(\mathcal{W} | \mathbf{x}(t)) p(\mathcal{A} | \mathbf{x}(t)). \quad (21)$$

The first term in (21) comprises the visual measurements only. Under the assumption that the measurements \mathbf{e}_k are independent of each other (given the trajectory and the map) and that the measurement error in the image coordinates of the events follows a zero-mean Gaussian distribution with variance σ_k^2 , we have

$$\log(p(\mathcal{E} | \mathbf{x}(t), \mathcal{M})) \quad (22)$$

$$= \log \left(\prod_k p(\mathbf{e}_k | \mathbf{x}(t_k), \mathcal{M}) \right) \quad (23)$$

$$= \log \left(\prod_k K_1 \exp \left(-\frac{\|\mathbf{e}_k - \hat{\mathbf{e}}_k(\mathbf{x}(t_k), \mathcal{M})\|^2}{2\sigma_k^2} \right) \right) \quad (24)$$

$$= \tilde{K}_1 - \frac{1}{2} \sum_k \frac{1}{\sigma_k^2} \|\mathbf{e}_k - \hat{\mathbf{e}}_k(\mathbf{x}(t_k), \mathcal{M})\|^2 \quad (25)$$

where $K_1 \doteq 1/\sqrt{2\pi\sigma_k^2}$ and $\tilde{K}_1 \doteq \sum_k \log K_1$ are constants (i.e., independent of the state $\mathbf{x}(t)$). Let us denote by $\hat{\mathbf{e}}_k(\mathbf{x}(t_k), \mathcal{M})$ the predicted value of the event location computed using the state $\mathbf{x}(t)$ and the map (18), \mathcal{M} . Such a prediction is a point on one of the projected 3-D primitives: in case of a map of points (17), $\hat{\mathbf{e}}$ is the projected point, and the norm in (25) is the standard reprojection error between two points; in case of a map of 3-D line segments (18), $\hat{\mathbf{e}}$ is a point on the projected line segment, and the norm in (25) is the Euclidean (orthogonal) distance from the observed point to the corresponding line segment [11]. In both cases, (i) the prediction is computed using the event camera trajectory at the time of the event, t_k , and (ii) we assume the data association to be known, i.e., the correspondences between events and map primitives.¹

¹In practice, we solve the data association using event-based pose-tracking algorithms that we run as a preprocessing step. Note that these algorithms rely only on the events. Details are provided in the experiments of Section VII.

Following similar steps as those in (22)-(25) (independence and Gaussian error assumptions), the second and third terms in (21) lead to

$$\log(p(\mathcal{W} | \mathbf{x}(t))) = \tilde{K}_2 - \frac{1}{2} \sum_i \frac{1}{\sigma_i^2} \|\boldsymbol{\omega}_i - \hat{\boldsymbol{\omega}}_i(\mathbf{x}(t_i))\|^2, \quad (26)$$

$$\log(p(\mathcal{A} | \mathbf{x}(t))) = \tilde{K}_3 - \frac{1}{2} \sum_j \frac{1}{\sigma_j^2} \|\mathbf{a}_j - \hat{\mathbf{a}}_j(\mathbf{x}(t_j))\|^2, \quad (27)$$

where $\hat{\boldsymbol{\omega}}_i$ and $\hat{\mathbf{a}}_j$ are predictions of the angular velocity and linear acceleration of the event camera computed using the modeled trajectory $\mathbf{x}(t)$, such as those given by (15) and (16) in the case of a cubic spline trajectory.

Collecting terms (25)-(27), the maximization of the likelihood (21), or equivalently, its logarithm, leads to the minimization of the objective function

$$F \doteq \sum_k \frac{1}{\sigma_k^2} \|\mathbf{e}_k - \hat{\mathbf{e}}_k(\mathbf{x}(t_k), \mathcal{M})\|^2 + \sum_i \frac{1}{\sigma_i^2} \|\boldsymbol{\omega}_i - \hat{\boldsymbol{\omega}}_i(\mathbf{x}(t_i))\|^2 + \sum_j \frac{1}{\sigma_j^2} \|\mathbf{a}_j - \hat{\mathbf{a}}_j(\mathbf{x}(t_j))\|^2, \quad (28)$$

where we omitted unnecessary constants. The first sum comprises the visual errors measured in the image plane and the last two sums comprise the inertial errors.

B. Constrained Optimization in Finite Dimensions

The objective function (28) is optimized with respect to the trajectory $\mathbf{x}(t)$ of the event camera, which in general is represented by an arbitrary curve in $SE(3)$, i.e., a “point” in an infinite-dimensional function space. However, because we represent the curve in terms of a finite set of known temporal basis functions (B-splines, formalized in (6)), the trajectory is parametrized by control poses $\mathbf{T}_{w,i}$ and, therefore, the optimization problem becomes finite dimensional. In particular, it is a non-linear least squares problem, for which standard numerical solvers such as Gauss-Newton or Levenberg-Marquardt can be applied.

In addition to the control poses, we optimize with respect to model parameters $\boldsymbol{\theta} = (\mathbf{b}_\omega^\top, \mathbf{b}_a^\top, s, \mathbf{o}^\top)^\top$, consisting of the IMU biases \mathbf{b}_ω and \mathbf{b}_a , and the map scale s and orientation with respect to the gravity direction \mathbf{o} . The map orientation \mathbf{o} is composed of roll and pitch angles, $\mathbf{o} = (\alpha, \beta)^\top$. Hence, we estimate the trajectory and additional model parameters by minimizing the objective function (28),

$$\{\mathbf{T}_{w,i}^*, \boldsymbol{\theta}^*\} = \arg \min_{\mathbf{T}, \boldsymbol{\theta}} F. \quad (29)$$

This optimization problem is solved in an iterative way using the Ceres solver [26], an efficient numerical implementation for non-linear least squares problems.

One final remark: the inertial predictions are computed as described in (15) and (16), using $\mathbf{T}_{w,s}$ and its derivatives, whereas the visual predictions require the computation of $\mathbf{T}_{w,s}^{-1}$. More specifically, for each event e_k , triggered at time t_k in the interval $[t_i, t_{i+1})$, we compute its pose $\mathbf{T}_{w,s}(u_k)$ using (6), where $u_k = (t_k - t_i)/\Delta t$. We then project the map point or line segment into the current image plane using projection matrices $\mathbf{P}(t_k) \sim \mathbf{K}(\mathbf{I}|\mathbf{0})\mathbf{T}_{w,s}^{-1}(t_k)$, \mathbf{K} being the time-invariant

intrinsic parameter matrix of the event camera (after radial distortion compensation), and compute the distance between the event location \mathbf{e}_k and the corresponding point $\hat{\mathbf{e}}_k$ in the projected primitive. To take into account the map scale s and orientation \mathbf{o} , we right-multiply $\mathbf{P}(t_k)$ by a similarity transformation with scale s and rotation $\mathbf{R}(\mathbf{o})$ before projecting the map primitives.

VII. EXPERIMENTS

We evaluate our method on several datasets using the two different map representations in Section V: lines-based maps and point-cloud based maps. These two representations allow us to evaluate the effect of two different visual error terms, which are the line-to-point distance and point-to-point reprojection errors presented in Section VI-A. In both cases, we quantify the trajectory accuracy using the ground truth of a motion-capture system. We use the same hand-eye calibration method as described in [27]. Having a monocular setup, the absolute scale is not observable from visual observations alone. However, we are able to estimate the absolute scale since the fused IMU measurements grant scale observability. The following two sections describe the experiments with line-based maps and point-based maps, respectively.

A. Trajectory Estimation in Line-based Maps

These experiments are similar to the ones presented in [11]. Here, however, we use the DAVIS instead of the DVS, which provides the following advantages. First, it has a higher spatial resolution of 240×180 pixels (instead of 128×128 pixels). Second, it provides inertial measurements (at 1 kHz) that are time-synchronized with the events. Third, it also outputs global-shutter intensity images (at 24 Hz) that we use for initialization, visualization, and a more accurate intrinsic camera calibration than that achieved using events.

1) *Tracking Method*: This method tracks a set of lines in a given metric map. Event-based line tracking is done using [19]. Pose estimation is then done using the Gold Standard PnP algorithm [23, p.181] on the intersection points of the lines. Fig. 4 shows tracking of two different shapes.

2) *Experiment*: We moved the DAVIS sensor by hand in a motion-capture system above a square pattern, as shown in Fig. 5a. The corresponding error plots in position and orientation are shown in Fig. 5b: position error is measured using the Euclidean distance, whereas orientation error is measured using the geodesic distance in $SO(3)$ (the angle of the relative rotation between the true rotation and the estimated one) [28]. The excitation in each degree of freedom and the corresponding six error plots are shown in Fig. 9. The error statistics are summarized in Table I.

We compare four algorithms against ground truth from a motion-capture system: (i) an image-based tracking algorithm (in green color in the figures), (ii) the event-based tracking algorithm in [19] (in cyan), (iii) the proposed spline-based optimization without IMU measurements (blue color), and (iv) the proposed spline-based optimization (with IMU measurements, in red color). As it can be seen in the figures and in Table I, the proposed spline-based optimization (“Spline

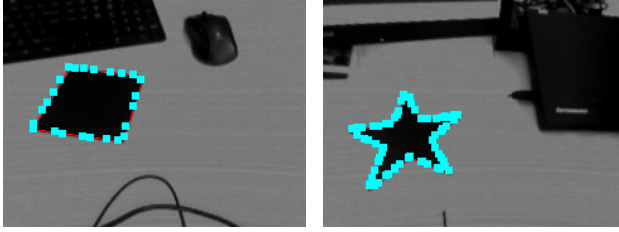
(ev. + IMU)” label) is more accurate than the event-based and the image tracking algorithms: the mean, standard deviation, and maximum errors in both position and orientation are the smallest among all methods (last row of Table I). The mean position error is 0.16 % of the average scene depth and the mean orientation error is 0.36° . The errors are approximately half of those of the second best method, the image-based tracker (cf. rows 1 and 4 in Table I). Hence, the proposed method is very accurate. The benefit of including the inertial measurements in the optimization is also reported: the vision-only spline-based optimization method is better than the event-based tracking algorithm [19] (by approximately a factor of 1.5) but it is worse than the image-based tracking algorithm (the error approximately doubles, cf. rows 1 and 3 in Table I). However, when inertial measurements are included in the optimization the errors are reduced by a factor of 4 approximately (by comparing rows 3 and 4 of Table I), hence the spline-based approach also outperforms the image-based tracker. Therefore, there is a significant gain in accuracy ($\times 4$ in this experiment) due to the fusion of inertial measurements and the event data to estimate the sensor’s trajectory.

For this experiment, we placed control poses at an interval of 0.1 s. This leads to ratios of about 5000 events and 100 inertial measurements per control pose. We initialize the control poses by fitting a spline trajectory through the initial tracker poses.

Scale Estimation: In further experiments, we also estimate the absolute scale s of the map as an additional parameter. As we know the map size precisely, we report the relative error. The square shape has a side length of 10 cm. For these experiments, we set the initial length to 0.1 cm, 1 cm, 1 m, and 10 m (two orders of magnitude in both directions). The optimization converged to virtually the same minimum and the relative error was below 7 % for all cases. This error is in the same ballpark as the magnitude error of the IMU, which we measured to be about 5 % (10.30 m/s^2 instead of 9.81 m/s^2 when the sensor is at rest).

B. Trajectory Estimation in Point-based Maps

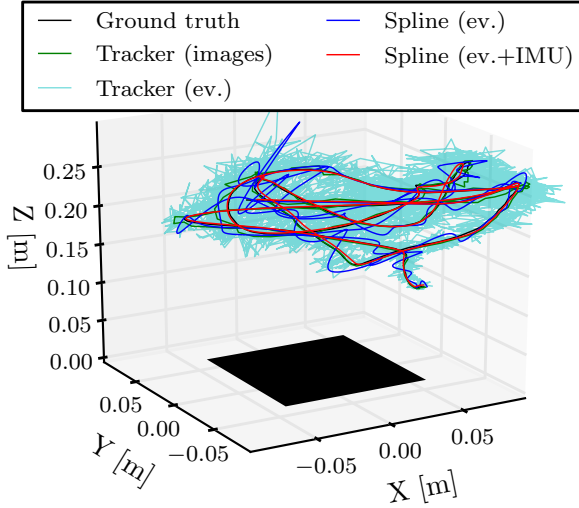
The following experiments show that the proposed continuous-time trajectory estimation also works on natural scenes, i.e., without requiring strong artificial gradients to generate the events. For this, we used three sequences from the Event-Camera Dataset [27], which we refer to as *desk*, *boxes* and *dynamic* (see Figs. 6a, 7a, and 8a). The *desk* scene features a desktop with some office objects (books, a screen, a keyboard, etc.); the *boxes* scene features some boxes on a carpet, and the *dynamic* scene consists of a desk with objects and a person moving them. All datasets were recorded hand-held and contain data from the DAVIS (events, frames, and inertial measurements) as well as ground-truth pose measurements from a motion capture system (at 200 Hz). We processed the data with EVO [12], an event-based visual-odometry algorithm, that we describe below, which also returns a point-cloud map of the scene. Then, we used the events and the point cloud map of EVO for batch trajectory optimization in the continuous-time framework, showing that we achieve higher accuracy and a smoother trajectory.



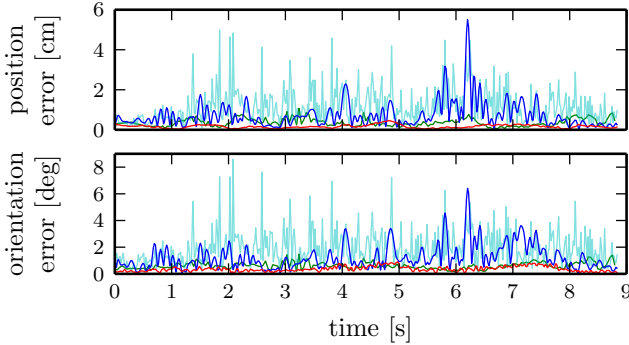
(a) Square shape.

(b) Star shape.

Fig. 4: Screenshots of the line-based tracking algorithm. The lines and the events used for its representation are in red and cyan, respectively. The image is only used for initialization and visualization.



(a) Estimated trajectories and 3D map.



(b) Trajectory error in position and orientation. Legend as in Fig. 5a.

Fig. 5: Results on Line-based Tracking and Pose Estimation.

TABLE I: Results on Line-based Tracking and Pose Estimation. Position and orientation errors.

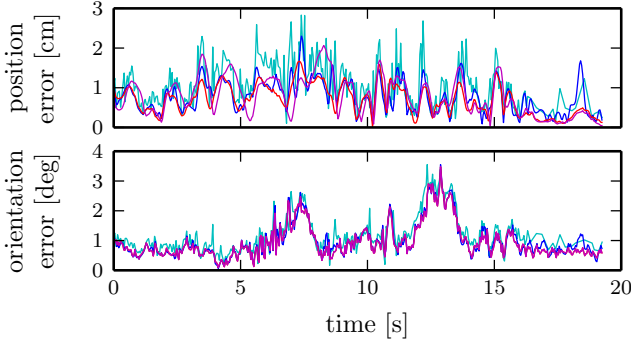
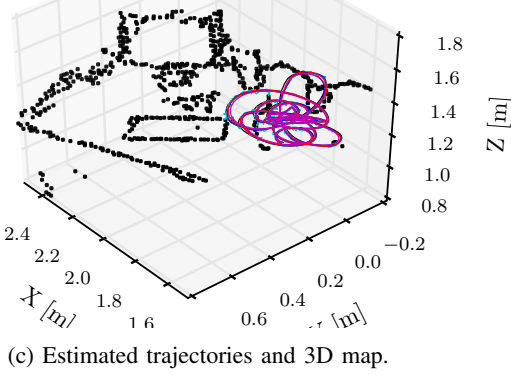
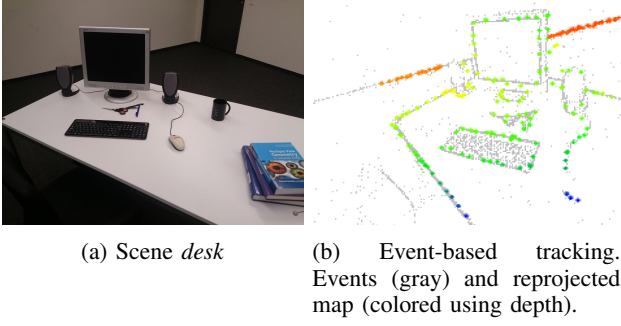
	Position error (abs. [cm] and rel. [%])						Orientation error [°]		
	μ	%	σ	%	max	%	μ	σ	max
Tracker (images)	0.36	1.06	0.20	0.58	1.07	3.15	0.61	0.25	1.48
Tracker (ev.) [19]	1.11	3.27	0.75	2.20	7.96	23.45	1.87	1.13	9.61
Spline (ev.)	0.73	2.15	0.65	1.90	5.53	16.27	1.24	0.91	6.45
Spline (ev.+IMU)	0.16	0.48	0.08	0.24	0.46	1.34	0.36	0.19	0.92

Relative errors are given with respect to the mean scene depth.

1) *Tracking Method*: EVO [12] returns both a map and a set of 6-DOF discrete, asynchronous poses of the event camera. In a post-processing step, we extracted the correspondences between the events and the map points that are required by our optimization. Figs. 6b, 7b, and 8b show typical point cloud maps produced by EVO, projected onto the image plane and colored according to depth with respect to the camera. The same plots also show all the observed events, colored in gray. Notice that the projected map is aligned with the observed events, as expected from an accurate tracking algorithm. The corresponding scenes are shown in Figs. 6a, 7a, and 8a, respectively.

2) *Experiments*: Figs. 6–8 and Tables II–IV summarize the results obtained on the three datasets. Fig. 6c, 7c, and 8c show the 3D maps and the event camera trajectories. Figs. 6d, 7d and 8d show the position and orientation errors obtained by comparing the estimated trajectories against motion-capture ground truth. Error statistics are provided in Tables II, III and IV. In additional plots in the Appendix (Figs. 9 to 12), we show the individual degrees of freedom and their errors, respectively.

We compare four methods against ground truth from the motion-capture system: (i) event-based pose tracking using EVO (in cyan color in the figures), (ii) spline-based trajectory optimization without IMU measurements (in blue color), (iii) spline-based trajectory optimization (events and inertial measurements, in red color), and (iv) spline-based trajectory and absolute scale optimization (in magenta). The output trajectory of each of the first three methods was aligned with respect to ground truth using a 3D similarity transformation (rotation, translation, and uniform scaling); thus, the absolute scale is externally provided. Although a Euclidean alignment suffices (rotation and translation, without scaling) for the spline-based approach with events and IMU, we also used a similarity alignment for a fair comparison with respect to other methods. The fourth method has the same optimized trajectory as the third one, but the alignment with respect to the ground truth trajectory is Euclidean (6-DOF): the absolute scale is recovered from the inertial measurements. As it can be seen in Tables II, III, and IV, the spline-based approach without inertial measurements consistently achieves smaller errors than EVO (cf. rows 1 and 2 of the tables). Using also the inertial measurements further improves the results (cf. rows 2 and 3 of the tables). When using the estimated absolute map scale, the results are comparable to those where the scale was provided by ground-truth alignment with a similarity transform, even though a low-cost IMU was used (cf. rows 3 and 4 of the tables). In such a case, the mean position error is less than 1.05 % of the average scene depth, and the mean orientation error is less than 1.03°. The standard deviations of the errors are also very small: less than 0.43 % and less than 0.57°, respectively, in all datasets. The results are remarkably accurate. The gain in accuracy due to incorporating the inertial measurements in the optimization (with respect to the visual-only approach) is less than a factor of two, which is not as large as in the case of line-based maps (a factor of four) because EVO [12] already provides very good results compared with the line-based tracker of [19]. Nevertheless, the gain is still

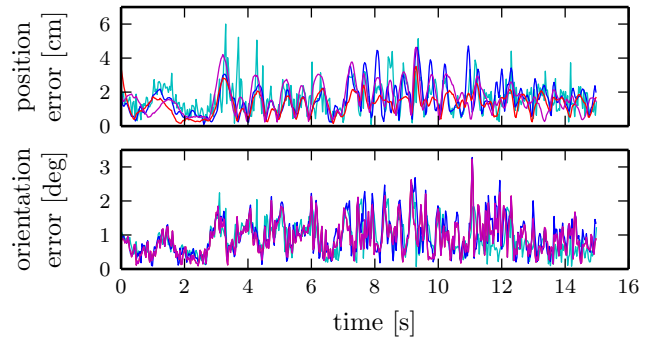
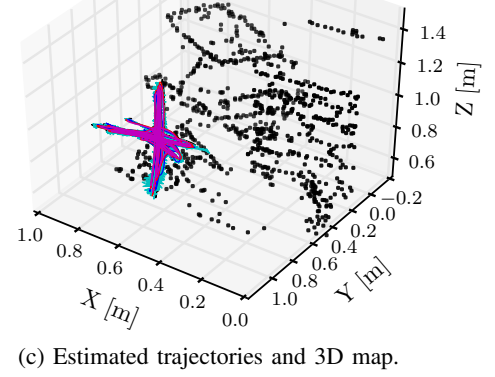
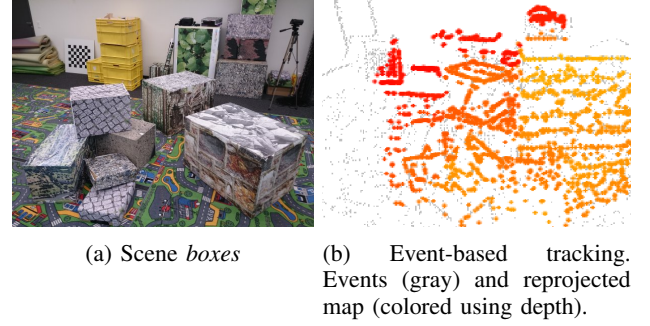
Fig. 6: Results for *desk* dataset.TABLE II: Results for *desk* dataset.

	Position error (abs. [cm] and rel. [%])						Orientation error [°]		
	μ	%	σ	%	max	%	μ	σ	max
EVO (ev.+abs. scale)	1.08	0.50	0.53	0.25	4.64	2.17	1.31	0.68	3.55
Spline (ev.+abs. scale)	0.78	0.36	0.40	0.19	2.30	1.08	0.98	0.58	3.56
Spline (ev.+IMU+abs. scale)	0.69	0.32	0.36	0.17	1.67	0.78	0.95	0.57	3.49
Spline (ev.+IMU)	0.78	0.36	0.47	0.22	2.07	0.97	0.95	0.57	3.49

Relative errors are given with respect to the mean scene depth.

significant, making the event-inertial optimization consistently outperforming the event-only one.

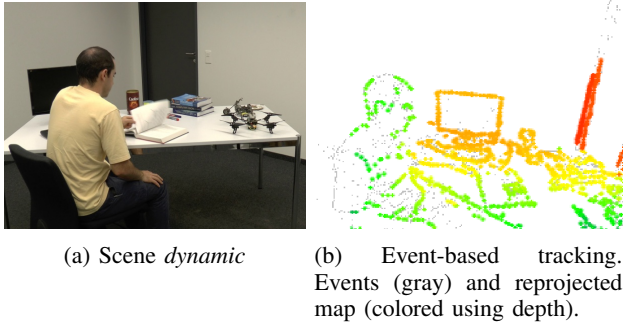
We placed the knots at a time interval of 0.2 s, 0.15 s, and 0.15 s for the *desk*, *boxes*, and *dynamic* datasets, respectively. This leads to a ratio of about 10^4 events and 150–200 inertial measurements per control pose. We initialize the control poses by fitting a spline through the initial tracker poses.

Fig. 7: Results for *boxes* dataset.TABLE III: Results for *boxes* dataset.

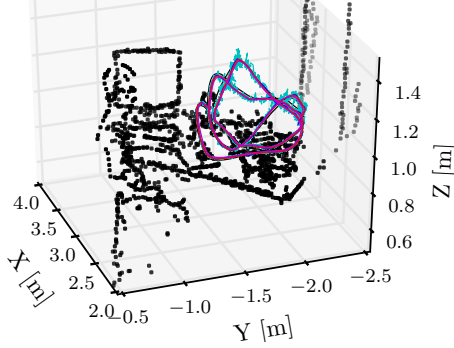
	Position error (abs. [cm] and rel. [%])						Orientation error [°]		
	μ	%	σ	%	max	%	μ	σ	max
EVO (ev.+abs. scale)	1.66	0.66	0.88	0.35	6.83	2.71	0.99	0.50	2.77
Spline (ev.+abs. scale)	1.58	0.63	0.89	0.35	4.72	1.87	0.99	0.54	3.28
Spline (ev.+IMU+abs. scale)	1.29	0.51	0.61	0.24	3.55	1.41	0.90	0.49	3.23
Spline (ev.+IMU)	1.65	0.65	0.87	0.35	4.68	1.86	0.90	0.49	3.23

Relative errors are given with respect to the mean scene depth.

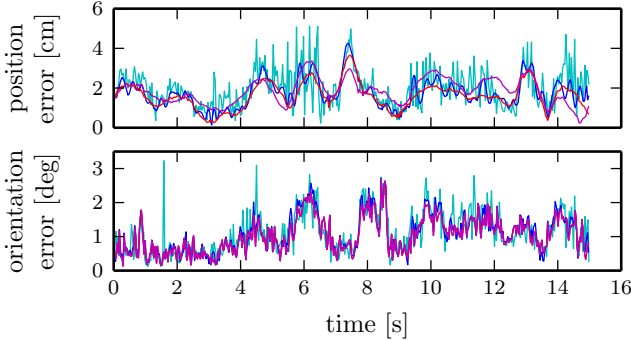
3) Absolute Map Scale and Gravity Alignment: In the above experiments with IMU, we also estimated the absolute scale s and orientation \mathbf{o} of the map as additional parameters. Since EVO is monocular, it cannot estimate the absolute scale. However, by fusing the inertial data with EVO, it is possible to recover the absolute scale and to align the map with gravity. We found that the absolute scale deviated from the true value by 4.1 %, 6.5 %, and 2.8 % for the *desk*, *boxes*, and *dynamic*

(a) Scene *dynamic*

(b) Event-based tracking. Events (gray) and reprojected map (colored using depth).



(c) Estimated trajectories and 3D map.



(d) Trajectory error in position and orientation. Legend as in (c).

Fig. 8: Results for *dynamic* dataset.TABLE IV: Results for *dynamic* dataset.

	Position error (abs. [cm] and rel. [%])						Orientation error [°]		
	μ	%	σ	%	max	%	μ	σ	max
EVO (ev.+abs. scale)	1.94	1.28	0.94	0.62	7.06	4.64	1.08	0.58	3.66
Spline (ev.+abs. scale)	1.74	1.14	0.74	0.49	6.50	4.27	1.08	0.56	3.42
Spline (ev.+IMU+abs. scale)	1.61	1.05	0.65	0.43	3.67	2.41	1.03	0.51	3.43
Spline (ev.+IMU)	1.85	1.21	0.64	0.42	3.36	2.21	1.03	0.51	3.43

Relative errors are given with respect to the mean scene depth.

datasets, respectively. For the alignment with gravity, we found that the estimated gravity direction deviated from the true value by 3.83° , 20.18° , and 3.34° for the *desk*, *boxes*, and *dynamic* datasets, respectively. The high alignment error for the *boxes* dataset is likely due to the dominant translational motion of the camera.

VIII. DISCUSSION

Event cameras provide visual measurements asynchronously and at very high rate. Traditional formulations, which describe the camera trajectory using poses at discrete timestamps, are not appropriate to deal with such almost continuous data streams because of the difficulty in establishing correspondences between the discrete sets of events and poses, and because the preservation of the temporal information of the events would require a very large number of poses (one per event). The continuous-time framework is a convenient representation of the camera trajectory since it has many desirable properties, among them: (i) it solves the issue of establishing correspondences between events and poses (since the pose at the time of the event is well-defined), and (ii) it is a natural framework for data fusion: it deals with the asynchronous nature of the events as well the synchronous samples from the IMU. As demonstrated in the experiments, such event-inertial data fusion allows significantly increasing the accuracy of the estimated camera motion over event-only-based approaches (e.g., by a factor of four).

The proposed parametric B-spline model makes the trajectory optimization computationally feasible since it has local basis functions (i.e., sparse Hessian matrix), analytical derivatives (i.e., fast to compute), and it is a compact representation: few parameters (control poses) suffice to assimilate several hundred thousand events and inertial measurements while providing a smooth trajectory. Additionally, batch optimization (simultaneous estimation of poses by exploiting their coupling), as opposed to filter-based approaches, is the preferred strategy to achieve maximum accuracy, at the expense of introducing some processing latency [29]. In this sense, our method demonstrated its usefulness to refine trajectories from state-of-the-art event-based pose trackers such as EVO, with or without inertial measurements. The current implementation runs off-line, as a post-processing stage, but the method can be adapted for on-line processing in a temporal sliding-window manner; the local support of the B-spline basis functions enables such type of local temporal processing.

Another reason for adopting the continuous-time trajectory framework is that it is agnostic to the map representation. We showed that the proposed method is flexible, capable of estimating accurate camera trajectories in scenes with line-based maps as well as point-cloud maps. In fact, the probabilistic (maximum likelihood) justification of the optimization approach gracefully unifies both formulations, lines and points, in the same objective function in a principled way. In this manner, we extended the method in [11] and broadened its applicability to different types of maps (i.e., scenes). Specifically, the results of the proposed method on line-based and point-based maps show similar remarkable accuracy, with mean position error of less than 1% of the average scene depth, and mean orientation error of less than 1° . Here, we assumed that, in the case of point-cloud maps, the data association between events and map points was given to focus on the optimization of the trajectory using visual-inertial data. The absolute scale and gravity direction are recovered in both types of maps, with an accuracy of approximately 5%, which matches the accuracy

of the IMU accelerometer; thus, the proposed method takes full advantage of the accuracy of the available sensor.

IX. CONCLUSION

In this paper, we presented a method to estimate the trajectory of an event camera using a continuous-time framework. This approach can deal with the asynchronous nature of the events and the high frequency of the inertial measurements in a principled way while providing a compact and smooth representation of the trajectory using a parametric cubic spline model. We optimized the approximated trajectory according to a geometrically meaningful error measure in the image plane and a direct inclusion of the inertial measurements, which have probabilistic justifications. We tested our method on real data from two recent algorithms: a simple line-based tracker and an event-based visual-odometry algorithm that works on natural scenes. In all experiments, our method outperformed previous algorithms when comparing to ground truth, with a remarkable accuracy: mean position error of less than 1 % of the average scene depth, and mean orientation error of less than 1° .

REFERENCES

- [1] A. Handa, R. A. Newcombe, A. Angeli, and A. J. Davison, "Real-time camera tracking: When is high frame-rate best?" in *Eur. Conf. Comput. Vis. (ECCV)*, 2012.
- [2] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128×128 120 dB 15 μ s latency asynchronous temporal contrast vision sensor," *IEEE J. Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, 2008.
- [3] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, "A 240×180 130dB 3 μ s latency global shutter spatiotemporal vision sensor," *IEEE J. Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, 2014.
- [4] L. Kneip, D. Scaramuzza, and R. Siegwart, "A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recog.*, 2011, pp. 2969–2976.
- [5] A. Patron-Perez, S. Lovegrove, and G. Sibley, "A spline-based trajectory representation for sensor fusion and rolling shutter cameras," *Int. J. Comput. Vis.*, vol. 113, no. 3, pp. 208–219, 2015.
- [6] C. Bibby and I. D. Reid, "A hybrid SLAM representation for dynamic marine environments," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2010, pp. 257–264.
- [7] P. Furgale, T. D. Barfoot, and G. Sibley, "Continuous-time batch estimation using temporal basis functions," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2012.
- [8] S. Anderson, F. Dellaert, and T. D. Barfoot, "A hierarchical wavelet decomposition for continuous-time SLAM," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2014.
- [9] H. S. Alismail, L. D. Baker, and B. Browning, "Continuous trajectory estimation for 3D SLAM from actuated lidar," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2014.
- [10] C. Kerl, J. Stückler, and D. Cremers, "Dense continuous-time tracking and mapping with rolling shutter RGB-D cameras," in *Int. Conf. Comput. Vis. (ICCV)*, 2015.
- [11] E. Mueggler, G. Gallego, and D. Scaramuzza, "Continuous-time trajectory estimation for event-based vision sensors," in *Robotics: Science and Systems (RSS)*, 2015.
- [12] H. Rebecq, T. Horstschäfer, G. Gallego, and D. Scaramuzza, "EVO: A geometric approach to event-based 6-DOF parallel tracking and mapping in real-time," *IEEE Robot. Autom. Lett.*, vol. 2, pp. 593–600, 2017.
- [13] T. Delbruck, V. Villanueva, and L. Longinotti, "Integration of dynamic vision sensor with inertial measurement unit for electronically stabilized event-based vision," in *IEEE Int. Symp. Circuits Syst. (ISCAS)*, Jun. 2014, pp. 2636–2639.
- [14] D. Weikersdorfer and J. Conradt, "Event-based particle filtering for robot self-localization," in *IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, 2012, pp. 866–870.
- [15] D. Weikersdorfer, R. Hoffmann, and J. Conradt, "Simultaneous localization and mapping for event-based vision systems," in *Int. Conf. Comput. Vis. Syst. (ICVS)*, 2013, pp. 133–142.
- [16] A. Censi and D. Scaramuzza, "Low-latency event-based visual odometry," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2014.
- [17] D. Weikersdorfer, D. B. Adrian, D. Cremers, and J. Conradt, "Event-based 3D SLAM with a depth-augmented dynamic vision sensor," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, Jun. 2014, pp. 359–364.
- [18] B. Kueng, E. Mueggler, G. Gallego, and D. Scaramuzza, "Low-latency visual odometry using event-based feature tracks," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, Daejeon, Korea, Oct. 2016, pp. 16–23.
- [19] E. Mueggler, B. Huber, and D. Scaramuzza, "Event-based, 6-DOF pose tracking for high-speed maneuvers," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2014, pp. 2761–2768.
- [20] G. Gallego, J. E. Lund, E. Mueggler, H. Rebecq, T. Delbruck, and D. Scaramuzza, "Event-based, 6-DOF camera tracking for high-speed applications," 2016, arXiv:1607.03468.
- [21] H. Kim, A. Handa, R. Benosman, S.-H. Ieng, and A. J. Davison, "Simultaneous mosaicing and tracking with an event camera," in *British Machine Vis. Conf. (BMVC)*, 2014.
- [22] H. Kim, S. Leutenegger, and A. J. Davison, "Real-time 3D reconstruction and 6-DoF tracking with an event camera," in *Eur. Conf. Comput. Vis. (ECCV)*, 2016, pp. 349–364.
- [23] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003, second Edition.
- [24] Y. Ma, S. Soatto, J. Košecká, and S. S. Sastry, *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer, 2004.
- [25] K. Qin, "General matrix representations for B-splines," *The Visual Computer*, vol. 16, no. 3–4, pp. 177–186, 2000.
- [26] A. Agarwal, K. Mierle, and Others, "Ceres solver," <http://ceres-solver.org>.
- [27] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza, "The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM," *Int. J. Robot. Research*, 2017.
- [28] D. Q. Huynh, "Metrics for 3D rotations: Comparison and analysis," *J. Math. Imaging Vis.*, vol. 35, no. 2, pp. 155–164, 2009.
- [29] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual-inertial odometry," *IEEE Trans. Robot.*, vol. 33, no. 1, pp. 1–21, Feb 2017.

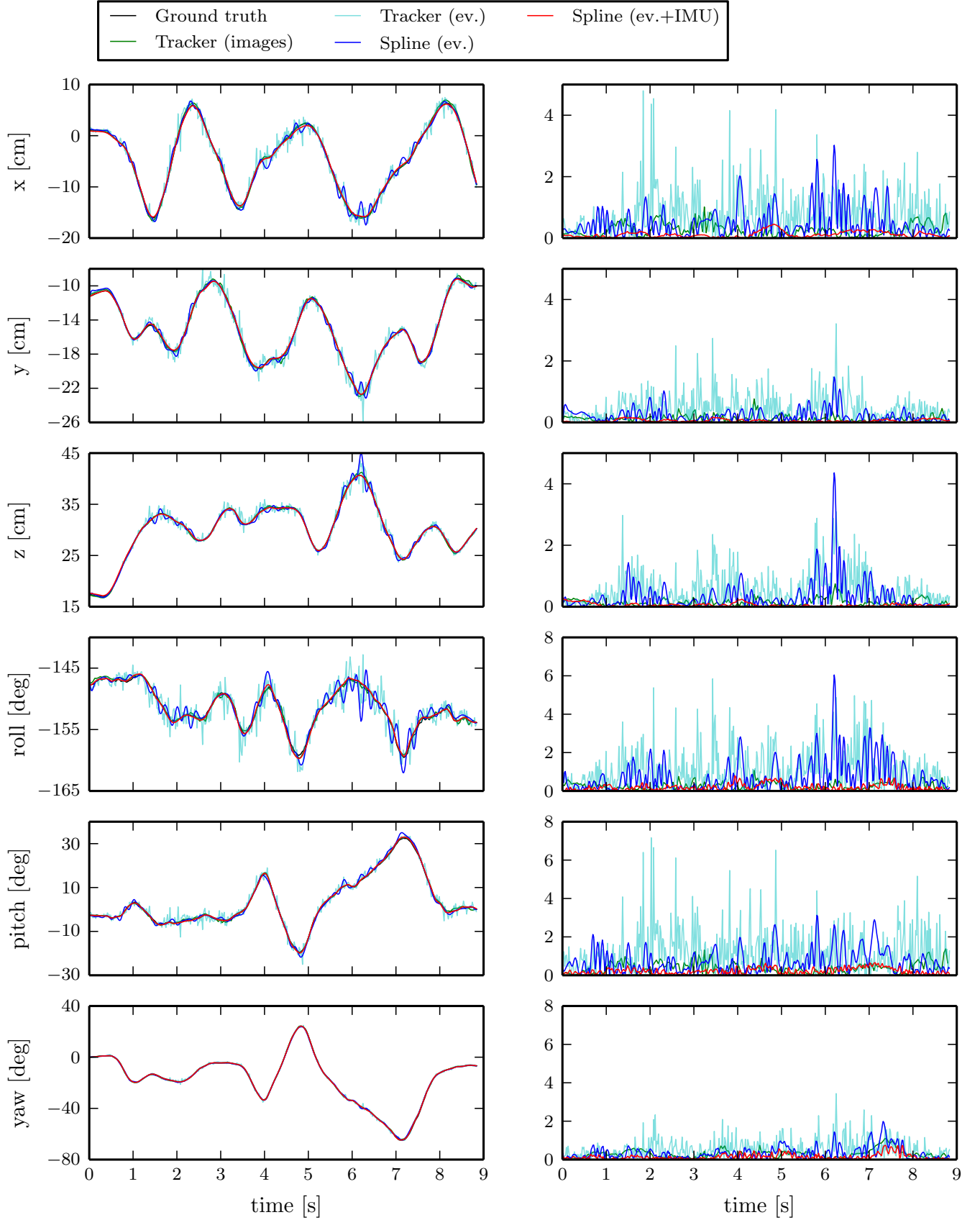


Fig. 9: *line-based* dataset. Plots of the 6-DOF (left column) and error (right column) of the estimated trajectories in Fig. 5a.

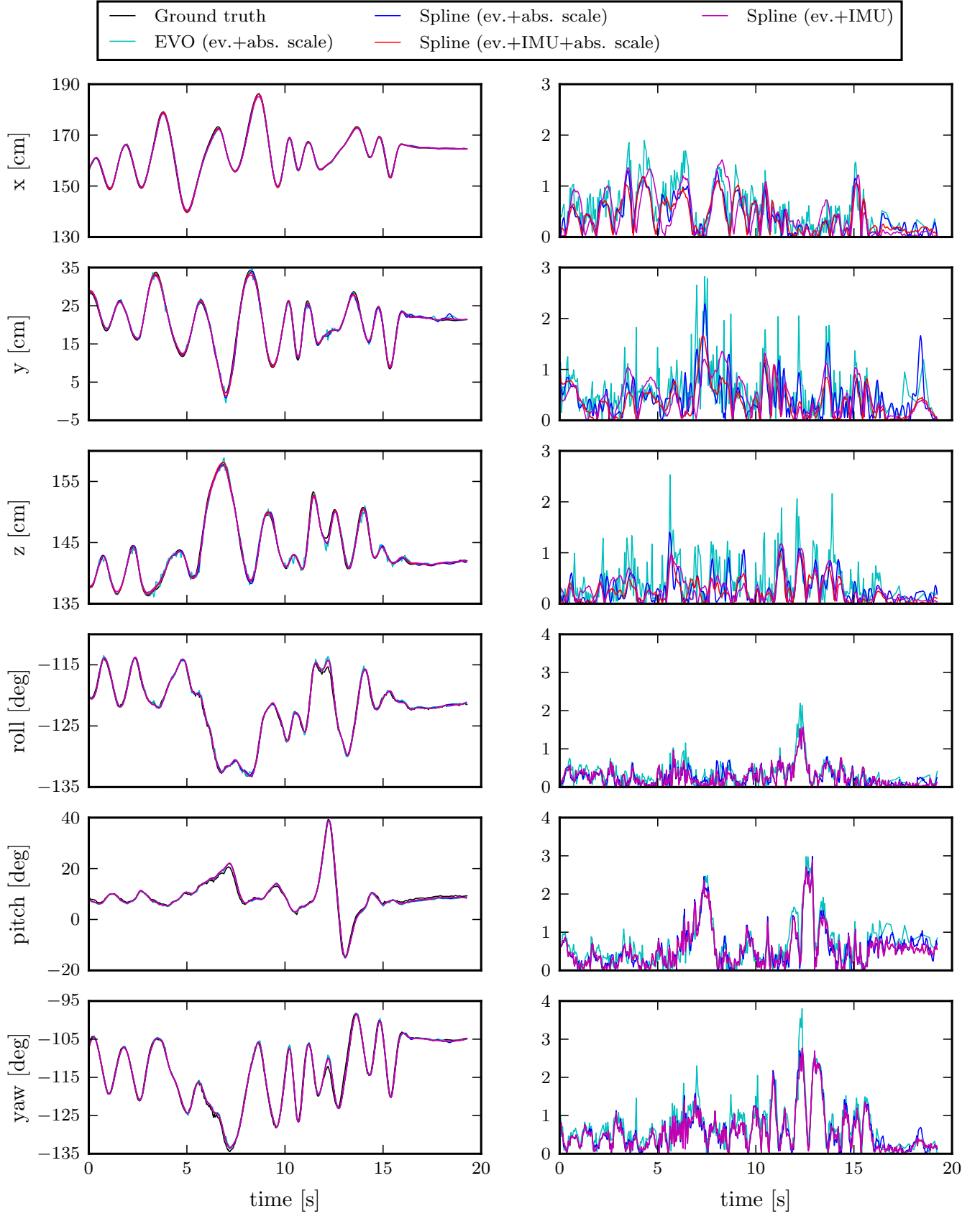


Fig. 10: *desk* dataset. Plots of the 6-DOF (left column) and error (right column) of the estimated trajectories in Fig. 6c.

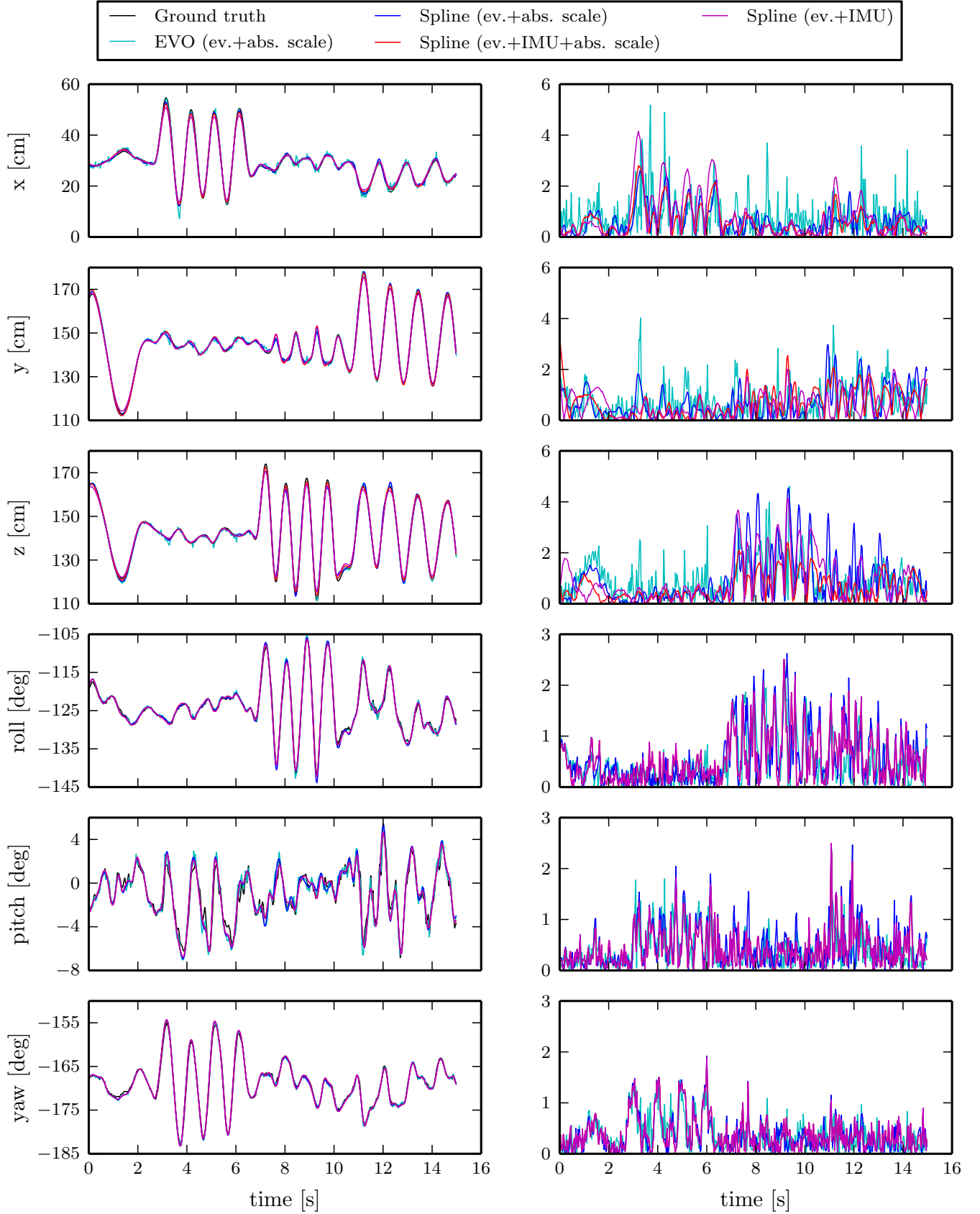


Fig. 11: *boxes* dataset. Plots of the 6-DOF (left column) and error (right column) of the estimated trajectories in Fig. 7c.

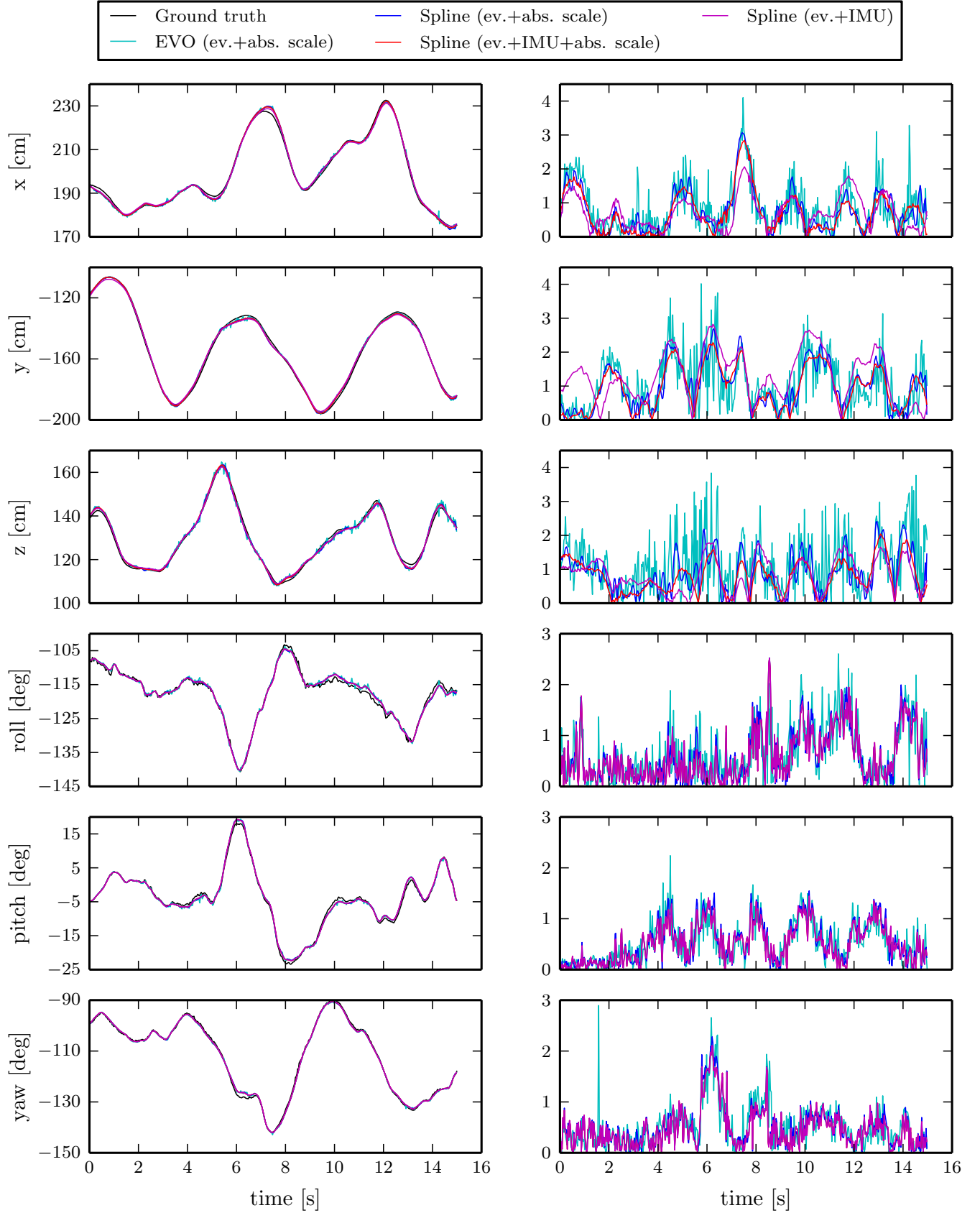


Fig. 12: *dynamic* dataset. Plots of the 6-DOF (left column) and error (right column) of the estimated trajectories in Fig. 8c